Optimization in the "Big Data" Regime 5: Parallelization?

Sham M. Kakade

Machine Learning for Big Data
CSE547/STAT548

University of Washington

Announcements...

- HW 3 posted
- Projects: the term is approaching the end.... (summer is coming!)

Today:

- Review: Adaptive gradient methods
- Parallelization: how do we parallelize in the “big data” regime?

Parallelization Overview

- Basic question: How can we do more operations “simultaneously” so that our overall task finishes more quickly?
- Issues:
  1. Break up computations on: single machine vs. cluster
  2. Breakup up: Models or Data
     Model parallelization or Data parallelization?
  3. Asynchrony?
- What are good models to study these issues?
  communication complexity, serial complexity, total computation, ??

1: One machine or a cluster?

- One machine:
  - Certain operations are much faster to do when specified without “for loops”: matrix multiplications, convolutions, Fourier transforms,
  - “parallelize” by structuring computations to take advantage of this: e.g. for larger matrix multiplies
  - GPUs!!!
  - Why one machine?
    Shared memory/communication is fast! Try to take advantage of fast “simultaneous” operations.
- Cluster:
  - Why? One machine can only do so much.
  - Try to (truly) breakup computations to be done.
  - Drawbacks: Communication is costly!
  - Simple method: run multiple jobs with different parameters.

2: Data Parallelization vs. Model parallelization

- Data parallelization:
  Breakup data into smaller chunks to process.
  - Mini-batching, batch gradient descent
  - Averaging
- Model parallelization:
  Breakup up your model.
  - Try to update parts of model in a distributed manner.
  - Coordinate ascent methods
  - Update layers of a neural net on different machines.
- Other issues:
  Asynchrony — e.g. Hogwild

Issues to consider...

- Work: the over all compute time.
- Depth: the serial runtime.
- Communication: between machines?
- Error: terminal error.
- Memory: how much do we need?

Mini-batching (Data Parallelization)

- stochastic gradient computation: at ~~stage s, using $\widetilde{w}_s$, compute~~:

$$w_{t+1} \leftarrow w_t - \eta \widehat{\nabla \ell(w_t)}$$

  where $\mathbb{E}\widehat{\nabla \ell(w)} = \nabla L(w)$.

- mini-batch SGD: using batch size b:

$$w_{t+1} \leftarrow w_t - \eta_b \left(\frac{1}{b} \sum_{j=1}^{b} \widehat{\nabla \ell_j(w_t)} \right)$$

  where η_b is our learning rate.

- How much does this help?
- It clearly reduces the variance.

How much does mini-batching help?

- Let's consider the regression/square loss case.
- $\overline{w}_t = \mathbb{E}[w_t]$ is the expected iterate at iteration t.
- Loss decomposition

$$L(w_t) - L(w_*) = \underbrace{L(w_t) - L(\overline{w}_t)}_{\text{Variance}} - \underbrace{(L(\overline{w}_t) - L(w_*))}_{\text{Bias}}$$

- mini-batch SGD: using batch size b:

$$w_{t+1} \leftarrow w_t - \eta_b \left(\frac{1}{b} \sum_{j=1}^{b} \widehat{\nabla \ell_j(w_t)} \right)$$

  where η_b is our learning rate.
- How much does this help?
  Bias? Variance?

Variance reduction with b?

- Initially: as long as the Bias $\geq$ Variance, then no point in variance reduction.
- Suppose you use b samples per SGD update (as opposed to $b = 1$). You run the same amount of time (as $b = 1$). How much does this help?
  - Error: Variance is always b times lower.
  - Depth: same
  - Work: b times more
- As $n \to \infty$ (with appropriate learning rate settings), the Bias$\to 0$ faster than the Variance. So mini-batching, in the limit, is helpful.
- How much does mini-batching help the bias?

Bias reduction with b?

- As we crank up $b \to \infty$, do we continue to expect improvements to the Bias?
  For large enough b, our update rule is:

$$w_{t+1} \leftarrow w_t - \eta \nabla L(w_t)$$

- What happens in between $b = 1$ and $b \to \infty$?
- Question: Let η_b^* be the maximal learning rate (again for the square loss case) that you can use before divergence. As you turn up b, how would we like η_b^* to change?

Example: sparse case

Example: sparse case

Example: sparse case

Bias reduction with b

- For $b = 1$,
$$\eta_1^* \approx \frac{1}{E[||x||^2]}$$
(it could be smaller for "heavy tailed" x).
- Lemma: As b increases, η_b increases and saturates to:
$$\eta_b^* \to \frac{1}{\lambda_{\max}}$$
- Let $\tilde{b}$ be this "critical point": the smallest b where $\eta_b^* \approx \frac{1}{\lambda_{\max}}$.
- (Informal) Theorem: Suppose you use $\tilde{b}$ samples per SGD update (as opposed to $b = 1$). You run the same amount of time (as $b = 1$).
  - η_b^* linearly increases until $\tilde{b}$.
  - Error: Bias contracts $\tilde{b}$ times as fast (per update).
  - Depth: same
  - Work: $\tilde{b}$ times more.
- So you can get to the same Bias in $\tilde{b}$ less depth (and the same amount of total work).

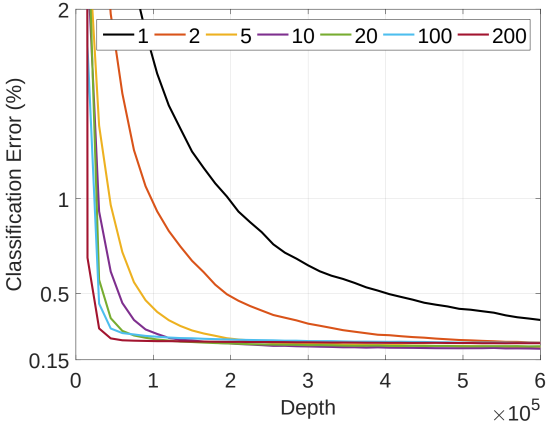
Putting it together: what is an optimal scheme?

- Initially: there is a "critical point" $\tilde{b}$ up to which mini-batching will help (both the bias and the variance).
- Informal Theorem: you can get to the same error in $\tilde{b}$ times less depth and the same total work (as compared to using $b = 1$).
- Practice: just crank up b until the error stops decreasing (as a function of the depth).
- Asymptotically (once the Bias becomes smaller than the variance), you could try to doubling tricks to increase the mini-batch size.
- Punchline: Unfortunately, $\tilde{b}$ usually will not be all that large for natural problems. This strongly favors the "GPU" model on one machine.

Is it general?

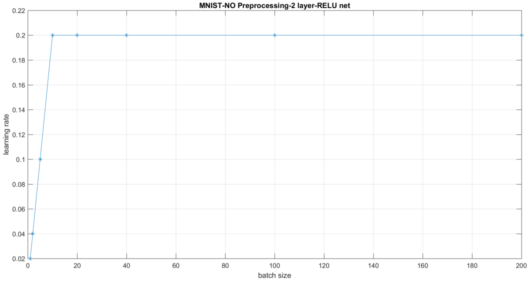
- Claims were only made for SGD in the square loss case.
- How general are these ideas? the non-convex case?
- Empirically, we often go with a “GPU model” where we max out our mini-batch size.

Mini-batching: Neural Net training on Mnist



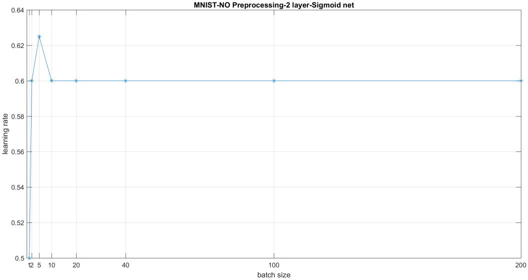
- Depth: The number of serial iterations.
- Work: Depth $\otimes$ the mini-batch size.

Neural Net Learning rates vs. batch size



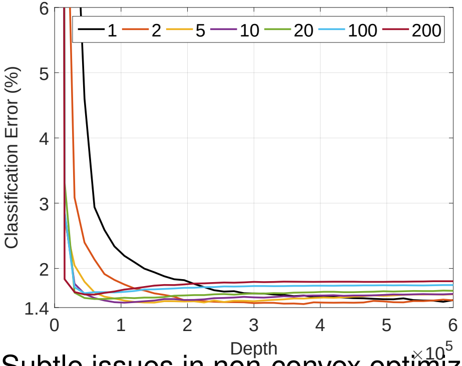
- The 'maximal' learning rate as a function of the batch size.
- Found with a crude grid search.

Neural Net Learning rates vs. batch size



- The 'maximal' learning rate as a function of the batch size.
- Found with a crude grid search.

Neural Net: Test error vs. batch size



- Subtle issues in non-convex optimization.
- More overfitting seen here with larger bath sizes.
  unclear how general this is.
- for this case, we were too aggressive with the learning rates.

Hogwild